F 5

```
BBBBBBBBBBBB        000000000        000000000     TTTTTTTTTTTTTTT     SSSSSSSSSSSS
BBBBBBBBBBBB        000000000        000000000     TTTTTTTTTTTTTTT     SSSSSSSSSSSS
BBBBBBBBBBBB        000000000        000000000     TTTTTTTTTTTTTTT     SSSSSSSSSSSS
BBB         BBB  000         000  000         000        TTT        SSS
BBB         BBB  000         000  000         000        TTT        SSS
BBB         BBB  000         000  000         000        TTT        SSS
BBB         BBB  000         000  000         000        TTT        SSS
BBB         BBB  000         000  000         000        TTT        SSS
BBB         BBB  000         000  000         000        TTT        SSS
BBBBBBBBBBBB     000         000  000         000        TTT           SSSSSSSSS
BBBBBBBBBBBB     000         000  000         000        TTT           SSSSSSSSS
BBBBBBBBBBBB     000         000  000         000        TTT           SSSSSSSSS
BBB         BBB  000         000  000         000        TTT                    SSS
BBB         BBB  000         000  000         000        TTT                    SSS
BBB         BBB  000         000  000         000        TTT                    SSS
BBB         BBB  000         000  000         000        TTT                    SSS
BBB         BBB  000         000  000         000        TTT                    SSS
BBB         BBB  000         000  000         000        TTT                    SSS
BBBBBBBBBBBB        000000000        000000000           TTT        SSSSSSSSSSSS
BBBBBBBBBBBB        000000000        000000000           TTT        SSSSSSSSSSSS
BBBBBBBBBBBB        000000000        000000000           TTT        SSSSSSSSSSSS
```

```
**FILE**ID**BOOTDRIVR                                          L 3







          BBBBBBBB      OOOOOO      OOOOOO    TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  RRRRRRRR
          BBBBBBBB      OOOOOO      OOOOOO    TTTTTTTTTT  DDDDDDDD    RRRRRRRR      IIIIII    VV      VV  RRRRRRRR
          BB      BB  OO      OO  OO      OO      TT      DD      DD  RR      RR      II      VV      VV  RR      RR
          BB      BB  OO      OO  OO      OO      TT      DD      DD  RR      RR      II      VV      VV  RR      RR
          BB      BB  OO      OO  OO      OO      TT      DD      DD  RR      RR      II      VV      VV  RR      RR
          BB      BB  OO      OO  OO      OO      TT      DD      DD  RR      RR      II      VV      VV  RR      RR
          BBBBBBBB    OO      OO  OO      OO      TT      DD      DD  RRRRRRRR        II      VV      VV  RRRRRRRR
          BBBBBBBB    OO      OO  OO      OO      TT      DD      DD  RRRRRRRR        II      VV      VV  RRRRRRRR
          BB      BB  OO      OO  OO      OO      TT      DD      DD  RR  RR          II      VV      VV  RR  RR
          BB      BB  OO      OO  OO      OO      TT      DD      DD  RR  RR          II      VV      VV  RR  RR
          BB      BB  OO      OO  OO      OO      TT      DD      DD  RR    RR        II        VV  VV    RR    RR        ....
          BB      BB  OO      OO  OO      OO      TT      DD      DD  RR    RR        II        VV  VV    RR    RR        ....
          BBBBBBBB      OOOOOO      OOOOOO        TT      DDDDDDDD    RR      RR    IIIIII        VV      RR      RR      ....
          BBBBBBBB      OOOOOO      OOOOOO        TT      DDDDDDDD    RR      RR    IIIIII        VV      RR      RR      ....

          LL            IIIIII      SSSSSSSS
          LL            IIIIII      SSSSSSSS
          LL              II      SS
          LL              II      SS
          LL              II      SS
          LL              II      SS
          LL              II        SSSSSS
          LL              II        SSSSSS
          LL              II              SS
          LL              II              SS
          LL              II              SS
          LL              II              SS
          LLLLLLLLLL    IIIIII    SSSSSSSS
          LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000        .TITLE  BOOTDRIVR DISPATCHER FOR BOOTSTRAP I/O DRIVERS
0000        .IDENT  'V04-000'
0000
0000  ;
0000  ;****************************************************************************
0000  ;*                                                                          *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000  ;*  ALL RIGHTS RESERVED.                                                    *
0000  ;*                                                                          *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000  ;*  TRANSFERRED.                                                            *
0000  ;*                                                                          *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000  ;*  CORPORATION.                                                            *
0000  ;*                                                                          *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000  ;*                                                                          *
0000  ;*                                                                          *
0000  ;****************************************************************************
0000  ;
0000  ;++
0000  ;
0000  ; FACILITY:
0000  ;
0000  ;       Minimal bootstrap driver for all VMS system disks.
0000  ;
0000  ; ENVIRONMENT:
0000  ;
0000  ;       Runs at IPL 31, kernel mode, memory management may be on or off,
0000  ;       IS=1 (running on interrupt stack), code must be PIC.
0000  ;
0000  ; ABSTRACT:
0000  ;
0000  ;       This module contains a routine called BOO$QIO that handles I/O
0000  ;       transfers to and from the VMS system disks.
0000  ;
0000  ; AUTHOR:
0000  ;
0000  ;       The VMS group
0000  ;
0000  ; REVISION HISTORY:
0000  ;
0000  ;       V03-011 TCM0005         Trudy C. Matthews       24-Jul-1984
0000  ;               Bump the VMB version number to indicate that the field
0000  ;               RPB$B_CTRLLTR is now being initialized.
0000  ;
0000  ;       V03-010 KPL0101         Peter Lieberwirth       11-Apr-1984
0000  ;               Update VMB version number for word-sized RPB field.  This
0000  ;               should have been done as part of v03-009.
0000  ;
```

```
0000   ;       V03-009 KPL0100         Peter Lieberwirth       12-Feb-1984
0000   ;               Change use of RPB$B_BOOTNDT to RPB$W_BOOTNDT, since BI
0000   ;               devices will have 16-bit device types.
0000   ;
0000   ;       V03-008 KDM0084         Kathleen D. Morse       23-Sep-1983
0000   ;               Add Micro-VAX I to CPUDISP.
0000   ;
0000   ;       V03-007 KDM0073         Kathleen D. Morse       22-Aug-1983
0000   ;               Add EXE$GL_TENUSEC and EXE$GL_UBDELAY to the fixed
0000   ;               data cells used by the bootstrap drivers.  Create
0000   ;               BOO symbols for these data cells.
0000   ;
0000   ;       V03-006 TCM0004         Trudy C. Matthews       02-Aug-1983
0000   ;               Add definition for EXE$GB_CPUDATA cell.
0000   ;
0000   ;       V03-005 KTA3059         Kerbey T. Altmann       21-Jun-1983
0000   ;               Add entries for unit disconnect and boot device name -
0000   ;               thus bumping VMB version number.
0000   ;
0000   ;       V03-004 RLRCPUDISP      Robert L. Rappaport     15-Jun-1983
0000   ;               Recode CPUDISP macros to use new format.
0000   ;
0000   ;       V03-003 TCM0003         Trudy C. Matthews       23-Feb-1983
0000   ;               Increment VMB version number to indicate adding RPB$L_BADPGS
0000   ;               field.
0000   ;
0000   ;       V03-002 TCM0002         Trudy C. Matthews       05-Jan-1983
0000   ;               Add 11/790-specific path to BOO$PURDPR.
0000   ;
0000   ;       V03-001 KTA0092         Kerbey T. Altmann       02-Apr-1982
0000   ;               Bump the version number because of KTA0090.
0000   ;
0000   ;       V02-021 KTA0090         Kerbey T. Altmann       26-Mar-1982
0000   ;               Add new cell to IOVEC to contain address of microcode
0000   ;               required by a booting device.
0000   ;
0000   ;--
```

```
          0000                .SBTTL  Declarations
          0000
          0000     ;
          0000     ; MACRO LIBRARY CALLS
          0000     ;
          0000
          0000                $BQODEF                             ; Define boot qio offsets
          0000                $BTDDEF                             ; Define boot device types
          0000                $IODEF                              ; DEFINE I/O FUNCTION CODES
          0000                $MBADEF                             ; DEFINE MASSBUS ADAPTER REGISTERS
          0000                $NDTDEF                             ; NEXUS device types
          0000                $PRDEF                              ; DEFINE PROCESSOR REGISTERS
          0000                $PTEDEF                             ; DEFINE PAGE TABLE ENTRY FIELDS
          0000                $RPBDEF                             ; DEFINE RESTART PARAMETER BLOCK
          0000                $SSDEF                              ; DEFINE STATUS CODES
          0000                $UBADEF                             ; UNIBUS ADAPTER REGISTER DEFINITIONS
          0000                $UBIDEF                             ; 11/750 UNIBUS adapter regs.
          0000                $VADEF                              ; DEFINE VIRTUAL ADDRESS FIELDS
          0000
          0000     ;
          0000     ; MACROS
          0000     ;
          0000
          0000     ;
          0000     ; LOCAL SYMBOLS
          0000     ;
          0000
          0000                $DEFINI BDT                         ; Define Boot Driver Table offsets
          0000
          0000     $DEF       BDT$L_CPUTYPE    .BLKW   1          ;  CPU type
          0002     $DEF       BDT$L_DEVTYPE    .BLKW   1          ;  Boot RO device type
          0004     $DEF       BDT$L_ACTION     .BLKL   1          ;  Action routine
          0008     $DEF       BDT$L_SIZE       .BLKL   1          ;  Driver size
          000C     $DEF       BDT$L_ADDR       .BLKL   1          ;  Driver address (offset)
          0010     $DEF       BDT$L_ENTRY      .BLKL   1          ;  Driver entry (offset from address)
          0014     $DEF       BDT$L_DRIVRNAME  .BLKL   1          ;  Driver name (offset from address)
          0018     $DEF       BDT$L_AUXDRNAME  .BLKL   1          ;  Auxiliary driver name (offset)
          001C     $DEF       BDT$L_UNIT_INIT  .BLKL   1          ;  Driver unit init (offset from address)
          0020     $DEF       BDT$L_UNIT_DISC  .BLKL   1          ;  Driver unit disc (offset from address)
          0024     $DEF       BDT$L_DEVNAME    .BLKL   1          ;  Device name (offset from address)
          0028
00000028  0028     BDT$K_LENGTH=.                                 ;  Length of entry
          0028
          0028                $DEFEND BDT                         ; End of Boot Driver Table definitions
```

```
                  0000          .SBTTL  DRIVER FIXED DATA AREA
                  0000
                  0000  ;
                  0000  ;       FIXED DATA CELLS FOR BOOTSTRAP DRIVER
                  0000  ;
                  0000
          00000000 0000         .PSECT  BOOTDRIVR_1,LONG        ; CERTAIN DRIVERS REQUIRE ALIGNMENT!
                  0000
                  0000  BOO$AL_VECTOR::                         ; VECTOR TO BOOT DRIVER ENTRY POINTS
         00000046' 0000         .LONG   BOO$QIO-BOO$AL_VECTOR   ; OFFSET TO BOOTSTRAP QIO ROUTINE
         0000011D' 0004         .LONG   BOO$MAP-BOO$AL_VECTOR   ; OFFSET TO MAPPING ROUTINE
         00000000' 0008         .LONG   BOO$SELECT-BOO$AL_VECTOR; OFFSET TO BOOTSTRAP I/O DRIVER
                  000C                                          ; INITIALLY SET TO ROUTINE WHICH
                  000C                                          ; SELECTS DRIVER
          00000000 000C         .LONG   0                       ; OFFSET TO SYSTEM DISK DRIVER NAME
                  0010                                          ; (ASCIC STRING). SET UP BY BOOT DRIVER.
                  0010  ;
                  0010  ; The next two words are the version number and the version number check fields.
                  0010  ; (The second word is the ones complement of the first word.)  The version
                  0010  ; number should be incremented whenever the interface between VMB and the
                  0010  ; rest of the system changes.  Release 1.0 VMB did not contain these fields.
                  0010  ;
                  0010  ;       Version 2 - Boot driver passes system disk driver name to SYSBOOT
                  0010  ;       Version 3 - VMB build memory description vector into RPB
                  0010  ;       Version 4 - VMB BOOTDRIVR purges UBA buffered datapath, all drivers
                  0010  ;                   return to BOOTDRIVR with success/failure status
                  0010  ;       Version 5 - VMB passes an argument list to the secondary boot
                  0010  ;                   in AP.  FILEREAD cacheing is present.
                  0010  ;       Version 6 - VMB passes nexus device type of boot adapter in
                  0010  ;                   RPB$B_BOOTNDT.
                  0010  ;       Version 7 - BOO$AL_VECTOR now has new entry points for RESELECTing
                  0010  ;                   a driver and UNIT_INIT for a driver.  Also new info
                  0010  ;                   passed in the argument list.
                  0010  ;       Version 8 - BOO$AL_VECTOR now has a new cell: BQO$L_UCODE.
                  0010  ;       Version 9 - VMB passes number of bad memory pages found during
                  0010  ;                   bootstrap scan in RPB$L_BADPGS.
                  0010  ;       Version 10- BOO$AL_VECTOR has two new cells: UNIT_DISC and DEVNAME
                  0010  ;
                  0010  ;       Version 11- BOO$AL_VECTOR has two new cells:  TENUSEC and UBDELAY
                  0010  ;
                  0010  ;       Version 12- RPB$W_BOOTNDT is defined, high byte of this word must
                  0010  ;                   be cleared in SYSBOOT for versions of VMB less than 12.
                  0010  ;
                  0010  ;       Version 13- RPB$B_CTRLLTR is defined; SYSBOOT must clear this field
                  0010  ;                   for older versions of VMB.
                  0010  ;
                  0010
          0000000D 0010 VMB_VERSION = 13
                  0010
                  0010          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$W_VERSION
         FFF2 000D 0010         .WORD   VMB_VERSION,^C<VMB_VERSION> ; VERSION # AND VERSION # CHECK FIELD.
         00000063' 0014         .LONG   BOO$RESELECT-BOO$AL_VECTOR ; Offset to set new driver
         00000012' 0018         .LONG   BOO$MOVE-BOO$AL_VECTOR  ; Offset to routine to select and move
                  001C          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UNIT_INIT
          00000000 001C         .LONG   0                       ; Offset to UNIT_INIT
                  0020          ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_AUXDRNAME
          00000000 0020         .LONG   0                       ; Offset to auxiliary driver name
          0024                                                      ;  second driver
          0024              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UMR_DIS
          0024   BOO$GL_UMR_DIS::                                   ; Number of map registers disabled
00000000  0024              .LONG   0
          0028              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UCODE
          0028   BOO$GL_UCODE::                                     ; Address of microcode in memory
00000000  0028              .LONG   0
          002C              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UNIT_DISC
00000000  002C              .LONG   0                               ; Offset to UNIT_DISC
          0030              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_DEVNAME
00000000  0030              .LONG   0                               ; Offset to boot device name
          0034              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UMR_TMPL
          0034   BOO$GL_UMR_TMPL::                                  ; UNIBUS map register template
80000000  0034              .LONG   UBA$M_MAP_VALID                 ;  Default is valid, no buff data path
          0038              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$B_UMR_DP
          0038   BOO$GB_UMR_DP::                                    ; UNIBUS map register data path
      01  0038              .BYTE   1                               ;  Default is Buffered #1
          0039              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$B_CPUTYPE
          0039   EXE$GB_CPUTYPE::                                   ; Location to hold processor
      01  0039              .BYTE   1                               ;  identification code
          003A              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_CPUDATA
          003A   EXE$GB_CPUDATA::                                   ; Location to hold contents of SID.
00000001  003A              .LONG   1
          003E              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_TENUSEC
          003E   EXE$GL_TENUSEC::                                   ; Location to hold TIMEDWAIT delay count
00000001  003E              .LONG   1
          0042              ASSUME  <.-BOO$AL_VECTOR> EQ BQO$L_UBDELAY
          0042   EXE$GL_UBDELAY::                                   ; Location to hold TIMEDWAIT delay count
00000001  0042              .LONG   1
```

```
                       0046         .SBTTL  BOO$QIO - BOOTSTRAP QIO ROUTINE
                       0046
                       0046 ;++
                       0046 ; FUNCTIONAL DESCRIPTION:
                       0046 ;
                       0046 ;       BOO$QIO PROVIDES THE DEVICE INDEPENDENT I/O INTERFACE FOR BOTH
                       0046 ;       READING AND WRITING THE BOOTSTRAP DEVICE.
                       0046 ;
                       0046 ; CALLING SEQUENCE:
                       0046 ;
                       0046 ;       CALLG   ARGLIST,BOO$QIO
                       0046 ;
                       0046 ; INPUT PARAMETERS:
                       0046 ;
                       0046 ;       BUF(AP)  - BUFFER ADDRESS
                       0046 ;       SIZE(AP) - SIZE OF BUFFER IN BYTES
                       0046 ;       LBN(AP)  - LOGICAL BLOCK NUMBER
                       0046 ;       FUNC(AP) - FUNCTION CODE
                       0046 ;                       ACCEPTS IO$_READLBLK AND IO$_WRITELBLK
                       0046 ;       MODE(AP) - ADDRESS INTERPRETATION MODE
                       0046 ;                       0 => PHYSICAL, 1 => VIRTUAL
                       0046 ;       RPB(AP) - ADDRESS OF RESTART PARAMETER BLOCK
                       0046 ;
                       0046 ; OUTPUT PARAMETERS:
                       0046 ;
                       0046 ;       R0 - COMPLETION STATUS CODE
                       0046 ;       R1 - TOTAL BYTES TRANSFERRED
                       0046 ;
                       0046 ;--
                       0046
                       0046 ;
                       0046 ; Offsets from AP to input arguments:
                       0046 ;
                       0046
             00000004  0046         BUF     = 4
             00000008  0046         SIZE    = 8
             0000000C  0046         LBN     = 12
             00000010  0046         FUNC    = 16
             00000014  0046         MODE    = 20
             00000018  0046         RPB     = 24
                       0046
                       0046 BOO$QIO::
                 0FFC  0046         .WORD   ^M<R2,R3,R4,R5,R6,R7,-   ; PRESERVE REGISTERS
                       0048                 R8,R9,R10,R11>
                       0048
                       0048 ;
                       0048 ; If mapping is enabled, the processor register RP$_MAPEN contains a 1.
                       0048 ; Otherwise, the register contains a 0. Use this value as an index to
                       0048 ; choose the appropriate address of the adapter's register space.
                       0048 ;
                       0048
     59   18 AC    D0  0048         MOVL    RPB(AP),R9              ; GET BASE ADDRESS OF RESTART PARAMETER BLK
        51    38   DB  004C         MFPR    #PR$_MAPEN,R1           ; CHECK FOR MAPPING ENABLED
                       004F         ASSUME  RPB$L_ADPVIR EQ RPB$L_ADPPHY+4
   53   5C A941    D0  004F         MOVL    RPB$L_ADPPHY(R9)[R1],R3 ; GET CORRECT POINTER TO CONFIG REG
                       0054
                       0054 ;
```

```
                         0054  ; Using the argument list as input, calculate the transfer size, number
                         0054  ; of map registers, starting LBN, starting VPN, and base of a page table
                         0054  ; to use in mapping.
                         0054  ;
                         0054
      5A   04 AC    D0   0054            MOVL    BUF(AP),R10                 ; Get buffer address
      58   08 AC    3C   0058            MOVZWL  SIZE(AP),R8                 ; GET TRANSFER SIZE IN BYTES
              04    12   005C            BNEQ    10$                         ; CONTINUE IF LEGAL SIZE
   58   01    10    9C   005E            ROTL    #16,#1,R8                   ; ELSE FORCE TO 64K SIZE
57   5A    09   00  EF   0062  10$:      EXTZV   #VA$V_BYTE,#VA$S_BYTE,R10,R7    ; Get byte offset into page
  57   03FF C847    9E   0067            MOVAB   ^X3FF(R8)[R7],R7            ; Calculate highest address plus
                         006D                                                ; an overflow page.
 57   57    F7 8F   78   006D            ASHL    #-9,R7,R7                   ; Reduce to number of pages
                         0072                                                ; (= number of map registers).
      5B   0C AC    D0   0072            MOVL    LBN(AP),R11                 ; AND BLOCK NUMBER FOR RETRY
      51   50 A9    D0   0076            MOVL    RPB$L_SVASPT(R9),R1         ; ASSUME SYSTEM SPACE
      03 5A   1F    E0   007A            BBS     #VA$V_SYSTEM,R10,20$        ; Branch if system address
         51   08    DB   007E            MFPR    #PR$_P0BR,R1                ; OTHERWISE GET P0 PT BASE
52   5A    15   09  EF   0081  20$:      EXTZV   #VA$V_VPN,#VA$S_VPN,R10,R2      ; Get base VPN for transfer
         66 A9      91   0086            CMPB    RPB$B_DEVTYP(R9),-          ; If booting from console block
              20         0089                    #BTD$K_HSCCI                ; storage device or CI,
              67    18   008A            BGEQ    PUSH_RETRY                  ; don't load map registers
                         008C
                         008C  ;
                         008C  ; Register usage right now is as follows:
                         008C  ;
                         008C  ;         R1      - address of page table for virtual-->physical mapping
                         008C  ;         R2      - base VPN for the transfer
                         008C  ;         R3      - address of the adapter's configuration register
                         008C  ;         R7      - number of map registers needed (plus one extra)
                         008C  ;         R8      - transfer size in bytes
                         008C  ;         R9      - address of the RPB
                         008C  ;         R10     - buffer address
                         008C  ;         R11     - starting LBN of the transfer
                         008C  ;
                         008C  ; In an adapter-dependent fashion, initialize the required number of
                         008C  ; adapter map registers. First calculate the address of the starting map
                         008C  ; register number. Right now, map registers for all UNIBUS and MASSBUS
                         008C  ; adapters for all processors start at the same offset from the base of
                         008C  ; the adapter's register space.
                         008C  ;
                         008C  ; During map register initialization, the following registers change
                         008C  ; for each page mapped:
                         008C  ;
                         008C  ;         R2      - address of the next VPN to map
                         008C  ;         R4      - address of the next map register to load
                         008C  ;         R5      - PFN of the page being mapped
                         008C  ;
                         008C  ;
                         008C  INIT_MAPREGS:                                 ; Initialize the map registers.
                         008C            ASSUME  MBA$L_MAP EQ UBA$L_MAP
                         008C            ASSUME  MBA$L_MAP EQ UBI$L_MAP
    54   FF94 CF    D0   008C            MOVL    W^BOO$GL_UMR_DIS,R4         ; Pick up number of disable UMR's
 54   0800 C344     DE   0091            MOVAL   MBA$L_MAP(R3)[R4],R4        ; Point to first useable map register
                         0097
                         0097  COMPUTE_PFN:                                  ; Loop once per page.
         55   82    9E   0097            MOVAB   (R2)+,R5                    ; Get a virtual page number.
```

```
             0B 14 AC   E9  009A               BLBC    MODE(AP),10$             ; If physical page #, branch.
            55    6145  D0  009E               MOVL    (R1)[R5],R5              ; Get page table entry
   55    FFE00000 8F    CA  00A2               BICL    #^C<PTE$M_PFN>,R5        ; and extract PFN from entry
                            00A9
                            00A9       ;
                            00A9       ; Derive the boot device adapter's type (UNIBUS adapter or MASSBUS
                            00A9       ; adapter) from the RPB, and save a flag indicating the adapter type
                            00A9       ; in a register.  The seemingly complicated fooling around with both
                            00A9       ; a UMR_DP and UMR_TMPL is to allow flexibility in what devices desire
                            00A9       ; in the way of data paths: Only UNIBUS devices will ever even pick up
                            00A9       ; the UMR_DP bit.  Thus all non-UNIBUS boot devices will never purge
                            00A9       ; a data path.  UNIBUS devices have a choice: by clearing UMR_DP in
                            00A9       ; their UNIT_INIT routines, the boot drivers can elect to not use
                            00A9       ; the buffered data path.
                            00A9       ;
                            00A9
                    50  D4  00A9       10$:    CLRL    R0                       ; Assume not UNIBUS
        56    00A1  C9  3C  00AB               MOVZWL  RPB$W_BOOTNDT(R9),R6     ; Pick up nexus type of boot adapter
              56    03  CA  00B0               BICL    #3,R6                    ; Make canonical adapter type
              28    56  B1  00B3               CMPW    R6, #NDT$_UBO            ; If boot adapter is a UNIBUS,
                    11  12  00B6               BNEQ    20$                      ;  then
                    50  D6  00B8               INCL    R0                       ;   Set a flag for later user
              FF7A  CF  F0  00BA               INSV    BOO$GB_UMR_DP,-          ;   Pick up the data path
              02    15      00BE                       #UBA$V_MAP_DPD,#2,-
              FF71  CF      00C0                       BOO$GL_UMR_TMPL          ;   and put it in the template
                            00C3
                            00C3       ;
                            00C3       ; This is a UNIBUS adapter.
                            00C3       ;
                            00C3       ; Map registers for the UNIBUS adapter look like the following:
                            00C3       ;
                            00C3       ;       +----------------------------------------------+
                            00C3       ;       | V |    | BO | DP # |   |  page frame number  |
                            00C3       ;       +----------------------------------------------+
                            00C3       ;
                            00C3       ; The code sets the byte offset bit if relevant, sets the valid bit,
                            00C3       ; sets the low bit in the 4-bit data path field to indicate that the
                            00C3       ; first buffered data path is to be used (if selected), and loads the
                            00C3       ; page frame number into the low bits.
                            00C3       ;
                            00C3
          19    04 AC   F0  00C3               INSV    BUF(AP),#UBA$V_MAP_BO,- ; Set UBA byte offset bit if
            55    01        00C7                       #1,R5                    ; necessary.
                            00C9       ;
                            00C9       ; ****** NOTE *****  For most devices, always uses Datapath #1,
                            00C9       ;                         IOC$PURDPR depends on this!!
                            00C9       ;
   84    FF66 CF    55  C9  00C9       20$:    BISL3   R5,BOO$GL_UMR_TMPL,(R4)+; Set PFN and byte offset, valid bit,
                            00CF                                                ;  and buffered DP number into map.
                            00CF       ;
                            00CF       ; This is a MASSBUS adapter.
                            00CF       ;
                            00CF       ; MASSBUS adapter's map registers look like the following:
                            00CF       ;
                            00CF       ;       +----------------------------------------+
                            00CF       ;       | V |                 | page frame number |
                            00CF       ;       +----------------------------------------+
                          00CF  ;
                          00CF  ;
                          00CF  ;*****
                          00CF  ;        BISL3   R5,#MBA$M_MAP_VALID,-   ; Set the PFN and the valid bit
                          00CF  ;                (R4)+                   ; in the map register.
                          00CF  ;*****
                          00CF  ;
                          00CF                                           ; Is there another page to do?
               57   D7    00CF           DECL    R7                      ; Decrement # of map registers.
               F4   14    00D1           BGTR    COMPUTE_PFN             ; Loop to fill next map register
                          00D3                                           ; if byte count not exhausted.
                          00D3
                          00D3  ;
                          00D3  ; All map registers are set up. Set up 2 more inputs to driver code.
                          00D3  ; Since the loaded MBA map registers are registers #0-n, the starting
                          00D3  ; address of the transfer to be loaded into a device register by the
                          00D3  ; device driver, is now simply the offset into the first page of the
                          00D3  ; transfer buffer.  However, for the UBA, there may be some disabled
                          00D3  ; map registers as a result of UNIBUS memory on the UBA.  Therefor,
                          00D3  ; the starting address of the transfer must include the lowest enabled
                          00D3  ; UMR in bits 9-17.
                          00D3  ;
                          00D3
           09   00   EF   00D3           EXTZV   #VA$V_BYTE,#VA$S_BYTE,- ; Get byte offset into page
           5A   5A        00D6                   R10,R10                 ; in R10
                          00D8
                          00D8  ;
                          00D8  ; Invalidate the last UBA map register so that a wild transfer will stop
                          00D8  ; at the end of the last valid block. Get the proper UMR to start with.
                          00D8  ;
                          00D8
           10   50   E9   00D8           BLBC    R0,30$                  ; Skip invalidation for MBA.
 74   80000000  8F   CA   00DB           BICL    #UBA$M_MAP_VALID,-(R4)  ; Invalidate last map register.
 50   FF3D CF   09   9C   00E2           ROTL    #9,W^BOO$GL_UMR_DIS,R0  ; Form UNIBUS address of UMR
           5A   50   C8   00E8           BISL    R0,R10                  ; Set into address register.
                          00EB
                          00EB  ;
                          00EB  ; If it is a UNIBUS boot device, derive the address of the device's CSR.
                          00EB  ;
                          00EB
                          00EB           ASSUME  RPB$L_CSRVIR  EQ  RPB$L_CSRPHY+4
                          00EB
           50   38   DB   00EB  30$:     MFPR    #PR$_MAPEN,R0           ; Check for mapping enabled
      57   54 A940   D0   00EE           MOVL    RPB$L_CSRPHY(R9)[R0],R7 ; Get address of device's CSR
                          00F3
                          00F3  PUSH_RETRY:
                0A   DD   00F3           PUSHL   #10                     ; Push retry count on stack
                          00F5
           55   5B   D0   00F5  10$:     MOVL    R11,R5                  ; Get a working copy of the block number
      50     34 A9   D0   00F8           MOVL    RPB$L_IOVEC(R9),R0      ; Get address of boot vectors
           08 B040   16   00FC           JSB     @BQO$L_SELECT(R0)[R0]   ; Call driver thru self-relative vector
                15   E1   0100           BBC     #UBA$V_MAP_DPD,-        ; Branch if not using the Buffered
        10 FF2F CF        0102                   BOO$GL_UMR_TMPL,100$    ;  data path
                50   DD   0106           PUSHL   R0                      ; Save driver status
              006A   30   0108           BSBW    BOO$PURDPR              ; Purge Buffered Datapath for UBA
             05 50   E8   010B           BLBS    R0,80$                  ; Branch if success
           5E   04   C0   010E           ADDL    #4,SP                   ; Clear previous status from stack
      06   11  0111         BRB     150$            ; Retry
      50 8ED0  0113  80$:   POPL    R0              ; Get driver status back
   03 50   E8  0116  100$:  BLBS    R0,200$         ; Branch if success
   D9 6E   F5  0119  150$:  SOBGTR  (SP),10$        ; Retry if count > 0
           04  011C  200$:  RET                     ; Return with final status in R0
```

```
                                 011D          .SBTTL  BOO$MAP - ROUTINE TO MAP DATA FOR BOO$QIO
                                 011D
                                 011D  ;++
                                 011D  ; FUNCTIONAL DESCRIPTION:
                                 011D  ;       BOO$MAP IS CALLED TO INITIALIZE THE DATA BASE FOR BOO$QIO TO PERMIT
                                 011D  ;       IT TO FUNCTION WITH MEMORY MANAGEMENT ENABLED.  AN AREA OF SYSTEM
                                 011D  ;       PAGE TABLE MUST BE PROVIDED SO THAT THE CONFIGURATION REGISTERS AND
                                 011D  ;       UNIBUS I/O PAGE CAN BE MAPPED.
                                 011D  ;
                                 011D  ; CALLING SEQUENCE:
                                 011D  ;       CALLG   ARGLIST,BOO$MAP
                                 011D  ;
                                 011D  ; INPUT PARAMETERS:
                                 011D  ;       SVASPT(AP) - SYSTEM VIRTUAL ADDRESS OF THE SYSTEM PAGE TABLE
                        00000004 011D          SVASPT  =    4
                                 011D  ;       VABASE(AP) - BASE VIRTUAL ADDRESS OF A 24 PAGE WINDOW TO MAP
                                 011D  ;                    THE ADAPTER CONFIGURATION REGISTERS AND UNIBUS
                        00000008 011D          VABASE  =    8
                                 011D  ;                    I/O PAGE.
                                 011D  ;       RPB(AP) -    ADDRESS OF RESTART PARAMETER BLOCK (RPB) CONTAINING
                                 011D  ;                    BOOTSTRAP DEVICE DESCRIPTION.
                        0000000C 011D          RPB     =    12
                                 011D  ;
                                 011D  ; OUTPUT PARAMETERS:
                                 011D  ;       NONE
                                 011D  ;
                                 011D  ;--
                                 011D
                            00FC 011D BOO$MAP::        .WORD   ^M<R2,R3,R4,R5,R6,R7>     ;
             57    0C AC      D0 011F          MOVL    RPB(AP),R7                ; GET BASE ADDRESS FOR RPB
             52    04 AC      D0 0123          MOVL    SVASPT(AP),R2             ; GET BASE OF SPT
             50 A7    52      D0 0127          MOVL    R2,RPB$L_SVASPT(R7)       ; AND SAVE IN DATA BASE
             53    08 AC      D0 012B          MOVL    VABASE(AP),R3             ; GET VIRTUAL ADDRESS OF WINDOW
             60 A7    53      D0 012F          MOVL    R3,RPB$L_ADPVIR(R7)       ; SET AS ADAPTER VIRTUAL ADDRESS
   54    5C A7    15    09    EF 0133          EXTZV   #VA$V_VPN,#VA$S_VPN,RPB$L_ADPPHY(R7),R4 ; GET BASE PFN
             55    08         D0 0139          MOVL    #8,R5                     ; SET TO MAP 8 PAGES
      50    53    15    09    EF 013C          EXTZV   #VA$V_VPN,#VA$S_VPN,R3,R0        ; GET BASE VIRTUAL PAGE
          51    6240          DE 0141          MOVAL   (R2)[R0],R1               ; COMPUTE WORKING SPT POINTER
                   20         10 0145          BSBB    FILLSPT                   ; FILL SPT TO MAP CONFIGURATION REGS
             55    10         D0 0147          MOVL    #16,R5                    ; SET FOR 16 PAGES
54    54 A7   00001FFF 8F     CB 014A          BICL3   #^X1FFF,RPB$L_CSRPHY(R7),R4      ; GET PHY ADDR OF I/O PAGE BASE
          54    54    17      9C 0153          ROTL    #<32-9>,R4,R4             ; AND CONVERT TO PAGE NUMBER
                      0E      10 0157          BSBB    FILLSPT                   ; STORE PTES INTO SPT
             50    54 A7      3C 0159          MOVZWL  RPB$L_CSRPHY(R7),R0       ; GET I/O PAGE OFFSET
   58 A7   FFFF3000 E043      9E 015D          MOVAB   <^X1000-^XE000>(R0)[R3],RPB$L_CSRVIR(R7) ; SET VIRTUAL CSR ADDR
                              04 0166          RET                               ;
                                 0167
                                 0167  ;++
                                 0167  ; FILLSPT
                                 0167  ;
                                 0167  ; INPUTS:
                                 0167  ;       R1 - POINTER TO CURRENT SPT ENTRY (UPDATED)
                                 0167  ;       R4 - PFN (UPDATED)
                                 0167  ;       R5 - COUNT OF PAGES TO FILL (UPDATED)
                                 0167  ;
                                 0167 FILLSPT:                                   ;
   81    54    90000000 8F    C9 0167          BISL3   #<PTE$M_VALID!PTE$C_KW>,R4,(R1)+ ; STORE A PTE
```

```
        54  D6  016F           INCL    R4                      ; ADVANCE TO NEXT PFN
     F3 55  F5  0171           SOBGTR  R5,FILLSPT              ; STORE THEM ALL
            05  0174           RSB
```

```
                           0175          .SBTTL  BOO$PURDPR - Purge UBA Buffered Datapath
                           0175
                           0175  ;++
                           0175  ; FUNCTIONAL DESCRIPTION:
                           0175  ;
                           0175  ;       This routine is called by BOOTDRIVR at the end of each boot device
                           0175  ;       transfer if the boot device is on the Unibus.  It purges the buffered
                           0175  ;       datapath and/or performs other Unibus adapter specific end-action.
                           0175  ;
                           0175  ;   NOTE: This routine contains processor specific code.
                           0175  ;
                           0175  ; CALLING SEQUENCE:
                           0175  ;
                           0175  ;       JSB     BOO$PURDPR
                           0175  ;
                           0175  ; INPUT PARAMETERS:
                           0175  ;
                           0175  ;       R3 - Address of UBA adapter configuration register
                           0175  ;       EXE$GB_CPUTYPE - Index specifying what CPU we are executing on
                           0175  ;       ** Assumes all drivers use DATAPATH 1 **
                           0175  ;
                           0175  ; OUTPUT PARAMETERS:
                           0175  ;
                           0175  ;       R0 - LBS -> Success
                           0175  ;            LBC -> Failure
                           0175  ;
                           0175  ;       R1,R2,R4 - Destroyed
                           0175  ;       All other registers preserved
                           0175  ;
                           0175  ;--
                           0175
                           0175  BOO$PURDPR:
                           0175
          50    01    3C   0175          MOVZWL  #SS$_NORMAL,R0             ; Assume success
                           0178          CPUDISP <<780,100$>,-              ; Dispatch on EXE$GB_CPUTYPE
                           0178                   <750,200$>,-
                           0178                   <730,300$>,-
                           0178                   <790,100$>,-
                           0178                   <UV1,170$>>,-              ; Nothing to do for Micro-VAX I
                           0178                  ENVIRON=VMB;
                           01AB
                           01AB  100$:                                      ; CPU type 11/780 and 11/790:
      52    44 A3    DE    01AB          MOVAL   UBA$L_DPR+4(R3),R2         ; Get Datapath Register address
   62    01    1F    78    01AF          ASHL    #UBA$V_DPR_BNE,#1,(R2)     ; Purge datapath
         51    62    D0    01B3          MOVL    (R2),R1                    ; Get Datapth register contents
      09 51    1E    E1    01B6          BBC     #UBA$V_DPR_XMTER,R1,170$   ; Branch if no error
   62    01    1E    78    01BA          ASHL    #UBA$V_DPR_XMTER,#1,(R2)   ; Clear error in datapath
   50  01F4    8F    3C    01BE  150$:   MOVZWL  #SS$_PARITY,R0             ; Set failure status
                     05    01C3  170$:   RSB                                ; Return to caller
                           01C4
      52    04 A3    DE    01C4  200$:   MOVAL   UBI$L_DPR+4(R3),R2         ; CPU type 11/750, Datapath Register
   62    01    00    78    01C8          ASHL    #UBI$V_DPR_PUR,#1,(R2)     ; Purge Datapath
         54    0A    D0    01CC          MOVL    #UBI$C_PURCNT,R4           ; Get max # of tries for
                           01CF                                             ; purge done test
         51    62    D0    01CF  230$:   MOVL    (R2),R1                    ; Get datapath register contents
      05 51    00    E1    01D2          BBC     #UBI$V_DPR_PUR,R1,250$     ; Branch if purge done
         F6 54       F5    01D6          SOBGTR  R4,230$                    ; Branch if more tries allowed
```

```
                 04     11  01D9               BRB     270$                   ; Return failure status
           E4 51  1F    E1  01DB      250$:    BBC     #UBI$V_DPR_ERROR,R1,170$ ; Branch if no purge error
              62  00    D2  01DF      270$:    MCOML   #0,(R2)                ; Clear datapath error(s)
                  DA    11  01E2               BRB     150$                   ; Return with failure status
                            01E4
           51    10 A3  D0  01E4      300$:    MOVL    UBI$L_SR(R3),R1        ; Get Unibus Error Summary Register
                            01E8                                              ; Nebula
   51      8001C000 8F  D3  01E8               BITL    #<UBI$M_SR_UWE!-       ; Any UB errors? (write error,
                            01EF                       UBI$M_SR_MRPE!-        ;       map parity error,
                            01EF                       UBI$M_SR_NXM!-         ;       non-existent memory,
                            01EF                       UBI$M_SR_UCE>,R1       ; or uncorrected read error.)
                  D2    13  01EF               BEQL    170$                   ; Branch if no errors
                            01F1      ; ****** QUESTION - Is there anything to do to clear the error status?
                  CB    11  01F1               BRB     150$                   ; Return failure status
                            01F3
              01F3          .ALIGN  LONG                   ; Alignment needed by some drivers!!!
    000001F4  01F4  BOO$QIOSIZ=.-BOO$AL_VECTOR             ; Size of boot QIO routine
              01F4
    000001F4  01F4  BOO$DRIVER==.                          ; Start of boot driver (after
              01F4                                         ; it's been moved)
              01F4                                         ; NOTE:  Boot drivers must be in
              01F4                                         ; psect BOOTDRIVR_2
              01F4
          00000000          .PSECT  BOOTDRIVR_3
              0000
    00000000  0000  BOO$DRIVER_TBL=.                       ; Boot driver table
              0000
          00000000          .PSECT  BOOTDRIVR_5
              0000
    00000000  0000          .LONG   0                      ; End of boot driver table
              0004
          00000000          .PSECT  BOOTDRIVR_6
```

```
                      0000          .SBTTL  BOO$SELECT - Select boot driver
                      0000
                      0000  ;++
                      0000  ; FUNCTIONAL DESCRIPTION:
                      0000  ;
                      0000  ;       This routine is called the first time BOO$QIO calls a driver.
                      0000  ;       It searches the boot driver table to locate the proper driver.
                      0000  ;       The correct linkage is made in BOO$AL_VECTOR.
                      0000  ;       RPB$L_IOVECSZ is also stored with the size of BOO$QIO plus
                      0000  ;       the size of the driver.  The driver is then jumped to.
                      0000  ;
                      0000  ; CALLING SEQUENCE:
                      0000  ;
                      0000  ;       JSB     BOO$SELECT      (Actually called through self-relative
                      0000  ;                                vector in BOO$AL_VECTOR+BQO$L_SELECT)
                      0000  ;
                      00C0  ; INPUT PARAMETERS:
                      0000  ;
                      0000  ;       R9      Address of the RPB
                      0000  ;
                      0000  ; OUTPUT PARAMETERS:
                      0000  ;
                      0000  ;       None
                      0000  ;
                      0000  ;--
                      0000
                      0000 BOO$SELECT:
     007E 8F    BB    0000          PUSHR   #^M<R1,R2,R3,R4,R5,R6>
          5D    10    0004          BSBB    BOO$RESELECT                    ; Select the correct driver
     007E 8F    BA    0006          POPR    #^M<R1,R2,R3,R4,R5,R6>
                      000A  ;
                      000A  ; Set up driver vector and jump to driver.
                      000A  ;
50    34 A9     D0    000A          MOVL    RPB$L_IOVEC(R9),R0              ; Get address of vectors
     08 B040    17    000E          JMP     @BQO$L_SELECT(R0)[R0]           ; Jump to driver
```

```
                          0012   650          .SBTTL  BOO$MOVE - Select and move boot driver
                          0012   651
                          0012   652 ;++
                          0012   653 ; FUNCTIONAL DESCRIPTION:
                          0012   654 ;
                          0012   655 ;        This routine is called after VMB is finished with a driver.
                          0012   656 ;        It searches the boot driver table to locate the proper driver.
                          0012   657 ;        The correct linkage is made in BOO$AL_VECTOR and driver moved.
                          0012   658 ;
                          0012   659 ; CALLING SEQUENCE:
                          0012   660 ;
                          0012   661 ;        JSB     BOO$MOVE        (Actually called through self-relative
                          0012   662 ;                                 vector in BOO$AL_VECTOR+BQO$L_MOVE)
                          0012   663 ;
                          0012   664 ; INPUT PARAMETERS:
                          0012   665 ;
                          0012   666 ;        R9      Address of the RPB
                          0012   667 ;
                          0012   668 ; OUTPUT PARAMETERS:
                          0012   669 ;
                          0012   670 ;        None
                          0012   671 ;
                          0012   672 ;--
                          0012   673
                          0012   674 BOO$MOVE:
      00FE 8F        BB   0012   675          PUSHR   #^M<R1,R2,R3,R4,R5,R6,R7> ; Save registers
              4B     10   0016   676          BSBB    BOO$RESELECT              ; Select the correct driver
   56   0C B545      9E   0018   677          MOVAB   @BDT$L_ADDR(R5)[R5],R6    ; Address of current position
   54   01F4'CF      9E   001D   678          MOVAB   W^BOO$DRIVER,R4           ; Address of new position
   57   56    54     C3   0022   679          SUBL3   R4,R6,R7                  ; Offset
              24     13   0026   680          BEQL    20$                       ; None, so don't move
64    66   08 A5     28   0028   681          MOVC3   BDT$L_SIZE(R5),(R6),(R4); Move driver
   54   0000'CF      9E   002D   682          MOVAB   W^BOO$AL_VECTOR,R4
      08 A4   57     C2   0032   683          SUBL2   R7,BQO$L_SELECT(R4)       ; Adjust offset
      0C A4   57     C2   0036   684          SUBL2   R7,BQO$L_DRIVRNAME(R4)
           20 A4     D5   003A   685          TSTL    BQO$L_AUXDRNAME(R4)       ; Is there one?
              04     13   003D   686          BEQL    10$                       ; No, don;t mess
      20 A4   57     C2   003F   687          SUBL2   R7,BQO$L_AUXDRNAME(R4)
           1C A4     D5   0043   688 10$:     TSTL    BQO$L_UNIT_INIT(R4)       ; Is there one?
              04     13   0046   689          BEQL    20$                       ; No, don;t mess
      1C A4   57     C2   0048   690          SUBL2   R7,BQO$L_UNIT_INIT(R4)
           2C A4     D5   004C   691 20$:     TSTL    BQO$L_UNIT_DISC(R4)       ; Is there one?
              04     13   004F   692          BEQL    30$                       ; No, don;t mess
      2C A4   57     C2   0051   693          SUBL2   R7,BQO$L_UNIT_DISC(R4)
           30 A4     D5   0055   694 30$:     TSTL    BQO$L_DEVNAME(R4)         ; Is there one?
              04     13   0058   695          BEQL    40$                       ; No, don;t mess
      30 A4   57     C2   005A   696          SUBL2   R7,BQO$L_DEVNAME(R4)
      00FE 8F        BA   005E   697 40$:     POPR    #^M<R1,R2,R3,R4,R5,R6,R7>
              05          0062   698          RSB
                          0063   699
                          0063   700 BOO$RESELECT:
   55   0000'CF      DE   0063   701          MOVAL   W^BOO$DRIVER_TBL,R5       ; Get address of boot driver table
    53     66 A9     9A   0068   702          MOVZBL  RPB$B_DEVTYP(R9),R3       ; Get value of boot device type
   54   0039'CF      9A   006C   703          MOVZBL  W^EXE$GB_CPUTYPE,R4       ; Get cpu type
   56   01F4'8F      3C   0071   704          MOVZWL  #<BOO$DRIVER-BOO$AL_VECTOR>,R6 ; Compute offset to driver table
                          0076   705 ;
                          0076   706 ; Determine if next driver in table is the correct one.
```

```
                         0076   ;
           50  65   32   0076   10$:    CVTWL   BDT$L_CPUTYPE(R5),R0    ; Get cpu type from table
               7B   13   0079           BEQL    400$                    ; End of table
               05   19   007B           BLSS    20$                     ; Driver doesn't care about cpu type
           54  50   D1   007D           CMPL    R0,R4                   ; Cpu types match?
               17   12   0080           BNEQ    40$                     ; No, try next driver
                         0082
        50  02 A5   32   0082   20$:    CVTWL   BDT$L_DEVTYPE(R5),R0    ; Get boot device type from table
               05   19   0086           BLSS    30$                     ; Driver doesn't care about device type
           53  50   D1   0088           CMPL    R0,R3                   ; Device types match?
               0C   12   008B           BNEQ    40$                     ; No, try next driver
                         008D
        50  04 A5   D0   008D   30$:    MOVL    BDT$L_ACTION(R5),R0     ; Get action routine offset from table
               0F   13   0091           BEQL    60$                     ; No action routine, this is the driver
             6540   16   0093           JSB     (R5)[R0]                ; Call action routine
            09 50   E8   0096           BLBS    R0,60$                  ; Branch if this is the driver
        56  08 A5   C0   0099   40$:    ADDL    BDT$L_SIZE(R5),R6       ; Account for this driver's size
           55  28   C0   009D           ADDL    #BDT$K_LENGTH,R5        ; Point to next driver entry
               D4   11   00A0           BRB     10$                     ; Try next driver
                         00A2   ;
                         00A2   ; Have the right driver.  R5 points to driver table entry. R6 contains
                         00A2   ; accumulated offset from IOVEC to the start of the driver.  Update
                         00A2   ; pertainent entries in the IOVEC.
                         00A2   ;
      54   0000'CF  DE   00A2   60$:    MOVAL   W^BOO$AL_VECTOR,R4      ; Cover the vector
       000001F4 8F  C1   00A7           ADDL3   #BOO$QIOSIZ,-           ; Add boot QIO size to
               08 A5     00AD                   BDT$L_SIZE(R5),-        ;  driver size
               38 A9     00AF                   RPB$L_IOVECSZ(R9)       ;  and store in RPB
       10 A5   56   C1   00B1           ADDL3   R6,BDT$L_ENTRY(R5),-    ; Calc offset to driver
               08 A4     00B5                   BQO$L_SECECT(R4)        ;  entry point and store in vector
       14 A5   56   C1   00B7           ADDL3   R6,BDT$L_DRIVRNAME(R5),-; Calc offset to driver
               0C A4     00BB                   BQO$L_DRIVRNAME(R4)     ;  name and store in vector
            1C A4   D4   00BD           CLRL    BQO$L_UNIT_INIT(R4)     ; Assume none
        51  1C A5   D0   00C0           MOVL    BDT$L_UNIT_INIT(R5),R1  ; Pick up possible UNIT_INIT entry
               05   13   00C4           BEQL    70$                     ; None specified, default to a RET
  1C A4    51  56   C1   00C6           ADDL3   R6,R1,BQO$L_UNIT_INIT(R4) ;Calc offset to driver
                         00CB                                           ;  UNIT_INIT point and store in vector
            20 A4   D4   00CB   70$:    CLRL    BQO$L_AUXDRNAME(R4)     ; Assume none
        51  18 A5   D0   00CE           MOVL    BDT$L_AUXDRNAME(R5),R1  ; Pick up possible driver name
               05   13   00D2           BEQL    80$                     ; None specified, default to a zero
  20 A4    51  56   C1   00D4           ADDL3   R6,R1,BQO$L_AUXDRNAME(R4) ; Calc offset to driver
                         00D9                                           ;  auxiliary name and store in vector
            2C A4   D4   00D9   80$:    CLRL    BQO$L_UNIT_DISC(R4)     ; Assume none
        51  20 A5   D0   00DC           MOVL    BDT$L_UNIT_DISC(R5),R1  ; Pick up possible UNIT DISC entry
               05   13   00E0           BEQL    90$                     ; None specified, default to a zero
  2C A4    51  56   C1   00E2           ADDL3   R6,R1,BQO$L_UNIT_DISC(R4) ; Calc offset to driver
                         00E7                                           ;  UNIT_DISC point and store in vector
            30 A4   D4   00E7   90$:    CLRL    BQO$L_DEVNAME(R4)       ; Assume none
        51  24 A5   D0   00EA           MOVL    BDT$L_DEVNAME(R5),R1    ; Pick up possible device name
               05   13   00EE           BEQL    100$                    ; None specified, default to a zero
  30 A4    51  56   C1   00F0           ADDL3   R6,R1,BQO$L_DEVNAME(R4) ; Calc offset to device
                         00F5                                           ;  name and store in vector
                    05   00F5   100$:   RSB
                         00F6
                         00F6   ;
                         00F6   ; No driver in the driver table accepted this QIO
                         00F6   ;
```

```
 00  00F6   764 400$:   HALT
     00F7   765
     00F7   766         .END
```

```
$$BASE                         = 00000001                FILLSPT                          00000167 R     02
$$DISPL                        = 00000008                FUNC                           = 00000010
$$GENSW                        = 00000001                INIT_MAPREGS                     0000008C R     02
$$HIGH                         = 00000007                LBN                            = 0000000C
$$LIMIT                        = 00000006                MBA$L_MAP                      = 00000800
$$LOW                          = 00000001                MODE                           = 00000014
$$MNSW                         = 00000001                NDT$_UB0                       = 00000028
$$MXSW                         = 00000001                OP$_ACBD                       = 0000006F
BDT$K_LENGTH                   = 00000028                OP$_ACBF                       = 0000004F
BDT$L_ACTION                     00000004                OP$_ACBG                       = 00004FFD
BDT$L_ADDR                       0000000C                OP$_ACBH                       = 00006FFD
BDT$L_AUXDRNAME                  00000018                OP$_ADDD2                      = 00000060
BDT$L_CPUTYPE                    00000000                OP$_ADDD3                      = 00000061
BDT$L_DEVNAME                    00000024                OP$_ADDF2                      = 00000040
BDT$L_DEVTYPE                    00000002                OP$_ADDF3                      = 00000041
BDT$L_DRIVRNAME                  00000014                OP$_ADDG2                      = 000040FD
BDT$L_ENTRY                      00000010                OP$_ADDG3                      = 000041FD
BDT$L_SIZE                       00000008                OP$_ADDH2                      = 000060FD
BDT$L_UNIT_DISC                  00000020                OP$_ADDH3                      = 000061FD
BDT$L_UNIT_INIT                  0000001C                OP$_ADDP4                      = 00000020
BOO$AL_VECTOR                    00000000 RG    02       OP$_ADDP6                      = 00000021
BOO$DRIVER                     = 000001F4 RG    02       OP$_ASHP                       = 000000F8
BOO$DRIVER_TBL                 = 00000000 R     03       OP$_CLRD                       = 0000007C
BOO$GB_UMR_DP                    00000038 RG    02       OP$_CLRF                       = 000000D4
BOO$GL_UCODE                     00000028 RG    02       OP$_CLRG                       = 0000007C
BOO$GL_UMR_DIS                   00000024 RG    02       OP$_CLRH                       = 00007CFD
BOO$GL_UMR_TMPL                  00000034 RG    02       OP$_CMPD                       = 00000071
BOO$MAP                          0000011D RG    02       OP$_CMPF                       = 00000051
BOO$MOVE                         00000012 R     05       OP$_CMPG                       = 000051FD
BOO$PURDPR                       00000175 R     02       OP$_CMPH                       = 000071FD
BOO$QIO                          00000046 RG    02       OP$_CMPP3                      = 00000035
BOO$QIOSIZ                     = 000001F4                OP$_CMPP4                      = 00000037
BOO$RESELECT                     00000063 R     05       OP$_CRC                        = 0000000B
BOO$SELECT                       00000000 R     05       OP$_CVTBD                      = 0000006C
BQO$B_CPUTYPE                  = 00000039                OP$_CVTBF                      = 0000004C
BQO$B_UMR_DP                   = 00000038                OP$_CVTBG                      = 00004CFD
BQO$L_AUXDRNAME                = 00000020                OP$_CVTBH                      = 00006CFD
BQO$L_CPUDATA                  = 0000003A                OP$_CVTDB                      = 00000068
BQO$L_DEVNAME                  = 00000030                OP$_CVTDF                      = 00000076
BQO$L_DRIVRNAME                = 0000000C                OP$_CVTDH                      = 000032FD
BQO$L_SELECT                   = 00000008                OP$_CVTDL                      = 0000006A
BQO$L_TENUSEC                  = 0000003E                OP$_CVTDW                      = 00000069
BQO$L_UBDELAY                  = 00000042                OP$_CVTFB                      = 00000048
BQO$L_UCODE                    = 00000028                OP$_CVTFD                      = 00000056
BQO$L_UHR_DIS                  = 00000024                OP$_CVTFG                      = 000099FD
BQO$L_UMR_TMPL                 = 00000034                OP$_CVTFH                      = 000098FD
BQO$L_UNIT_DISC                = 0000002C                OP$_CVTFL                      = 0000004A
BQO$L_UNIT_INIT                = 0000001C                OP$_CVTFW                      = 00000049
BQO$W_VERSION                  = 00000010                OP$_CVTGB                      = 000048FD
BTD$K_HSCCI                    = 00000020                OP$_CVTGF                      = 000033FD
BUF                            = 00000004                OP$_CVTGH                      = 000056FD
COMPUTE_PFN                      00000097 R     02       OP$_CVTGL                      = 00004AFD
ERROUT                           ********   X   02       OP$_CVTGW                      = 000049FD
EXE$GB_CPUDATA                   0000003A RG    02       OP$_CVTHB                      = 000068FD
EXE$GB_CPUTYPE                   00000039 RG    02       OP$_CVTHD                      = 0000F7FD
EXE$GL_TENUSEC                   0000003E RG    02       OP$_CVTHF                      = 0000F6FD
EXE$GL_UBDELAY                   00000042 RG    02       OP$_CVTHG                      = 000076FD
```

```
OP$_CVTHL                   = 00006AFD          OP$_POLYG                   = 000055FD
OP$_CVTHW                   = 000069FD          OP$_POLYH                   = 000075FD
OP$_CVTLD                   = 0000006E          OP$_SCANC                   = 0000002A
OP$_CVTLF                   = 0000004E          OP$_SKPC                    = 0000003B
OP$_CVTLG                   = 00004EFD          OP$_SPANC                   = 0000002B
OP$_CVTLH                   = 00006EFD          OP$_SUBD2                   = 00000062
OP$_CVTLP                   = 000000F9          OP$_SUBD3                   = 00000063
OP$_CVTPL                   = 00000036          OP$_SUBF2                   = 00000042
OP$_CVTPS                   = 00000008          OP$_SUBF3                   = 00000043
OP$_CVTPT                   = 00000024          OP$_SUBG2                   = 000042FD
OP$_CVTRDL                  = 0000006B          OP$_SUBG3                   = 000043FD
OP$_CVTRFL                  = 0000004B          OP$_SUBH2                   = 000062FD
OP$_CVTRGL                  = 00004BFD          OP$_SUBH3                   = 000063FD
OP$_CVTRHL                  = 00006BFD          OP$_SUBP4                   = 00000022
OP$_CVTSP                   = 00000009          OP$_SUBP6                   = 00000023
OP$_CVTTP                   = 00000026          OP$_TSTD                    = 00000073
OP$_CVTWD                   = 0000006D          OP$_TSTF                    = 00000053
OP$_CVTWF                   = 0000004D          OP$_TSTG                    = 000053FD
OP$_CVTWG                   = 00004DFD          OP$_TSTH                    = 000073FD
OP$_CVTWH                   = 00006DFD          PR$_MAPEN                   = 00000038
OP$_DIVD2                   = 00000066          PR$_POBR                    = 00000008
OP$_DIVD3                   = 00000067          PR$_SID_TYP730              = 00000003
OP$_DIVF2                   = 00000046          PR$_SID_TYP750              = 00000002
OP$_DIVF3                   = 00000047          PR$_SID_TYP780              = 00000001
OP$_DIVG2                   = 000046FD          PR$_SID_TYP790              = 00000004
OP$_DIVG3                   = 000047FD          PR$_SID_TYPUV1              = 00000007
OP$_DIVH2                   = 000066FD          PTE$C_KW                    = 10000000
OP$_DIVH3                   = 000067FD          PTE$M_PFN                   = 001FFFFF
OP$_DIVP                    = 00000027          PTE$M_VALID                 = 80000000
OP$_EDITPC                  = 00000038          PUSH_RETRY                    000000F3 R      02
OP$_EMODD                   = 00000074          RPB                         = 0000000C
OP$_EMODF                   = 00000054          RPB$B_DEVTYP                = 00000066
OP$_EMODG                   = 000054FD          RPB$L_ADPPHY                = 0000005C
OP$_EMODH                   = 000074FD          RPB$L_ADPVIR                = 00000060
OP$_MATCHC                  = 00000039          RPB$L_CSRPHY                = 00000054
OP$_MNEGD                   = 00000072          RPB$L_CSRVIR                = 00000058
OP$_MNEGF                   = 00000052          RPB$L_IOVEC                 = 00000034
OP$_MNEGG                   = 000052FD          RPB$L_IOVECSZ               = 00000038
OP$_MNEGH                   = 000072FD          RPB$L_SVASPT                = 00000050
OP$_MOVD                    = 00000070          RPB$W_BOOTNDT               = 000000A1
OP$_MOVF                    = 00000050          SIZE                        = 00000008
OP$_MOVG                    = 000050FD          SS$_NORMAL                  = 00000001
OP$_MOVH                    = 000070FD          SS$_PARITY                  = 000001F4
OP$_MOVP                    = 00000034          SVASPT                      = 00000004
OP$_MOVTC                   = 0000002E          UBA$L_DPR                   = 00000040
OP$_MOVTUC                  = 0000002F          UBA$L_MAP                   = 00000800
OP$_MULD2                   = 00000064          UBA$M_MAP_VALID             = 80000000
OP$_MULD3                   = 00000065          UBA$V_DPR_BNE               = 0000001F
OP$_MULF2                   = 00000044          UBA$V_DPR_XMTER             = 0000001E
OP$_MULF3                   = 00000045          UBA$V_MAP_BO                = 00000019
OP$_MULG2                   = 000044FD          UBA$V_MAP_DPD               = 00000015
OP$_MULG3                   = 000045FD          UBI$C_PURCNT                = 0000000A
OP$_MULH2                   = 000064FD          UBI$L_DPR                   = 00000000
OP$_MULH3                   = 000065FD          UBI$L_MAP                   = 00000800
OP$_MULP                    = 00000025          UBI$L_SR                    = 00000010
OP$_POLYD                   = 00000075          UBI$M_SR_MRPE               = 00008000
OP$_POLYF                   = 00000055          UBI$M_SR_NXM                = 00010000
```

```
UBI$M_SR_UCE                    = 80000000
UBI$M_SR_UWE                    = 00004000
UBI$V_DPR_ERROR                 = 0000001F
UBI$V_DPR_PUR                   = 00000000
VA$S_BYTE                       = 00000009
VA$S_VPN                        = 00000015
VA$V_BYTE                       = 00000000
VA$V_SYSTEM                     = 0000001F
VA$V_VPN                        = 00000009
VABASE                          = 00000008
VMB_VERSION                     = 0000000D

                                             +----------------+
                                             ! Psect synopsis !
                                             +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           00000028  (    40.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
BOOTDRIVR_1                     000001F4  (   500.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG
BOOTDRIVR_3                     00000000  (     0.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
BOOTDRIVR_5                     00000004  (     4.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
BOOTDRIVR_6                     000000F7  (   247.)  05 (  5.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC BYTE

                                          +------------------------+
                                          ! Performance indicators !
                                          +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     29    00:00:00.09    00:00:00.33
Command processing                108    00:00:00.80    00:00:02.74
Pass 1                            627    00:00:23.76    00:00:48.10
Symbol table sort                   0    00:00:02.72    00:00:05.54
Pass 2                            161    00:00:05.84    00:00:11.03
Symbol table output                28    00:00:00.23    00:00:00.78
Psect synopsis output               3    00:00:00.03    00:00:00.04
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              958    00:00:33.47    00:01:08.57

The working set limit was 2000 pages.
114533 bytes (224 pages) of virtual memory were used to buffer the intermediate code.
There were 100 pages of symbol table space allocated to hold 1738 non-local and 39 local symbols.
3518 source lines were read in Pass 1, producing 20 object records in Pass 2.
157 pages of virtual memory were used to define 154 macros.
```

```
                                              +---------------------------+
                                              ! Macro library statistics !
                                              +---------------------------+

Macro library name                            Macros defined
------------------                            --------------
_$255$DUA28:[BOOTS.OBJ]BOOTS.MLB;1                    0
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       11
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     8
TOTALS (all libraries)                               19

1889 GETS were required to define 19 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:BOOTDRIVR/OBJ=OBJ$:BOOTDRIVR MASD$:[EMULAT.SRC]MISSING/UPDATE=(MASD$:[EMULAT.ENH]MISSING)+MASD$:[BOOTS.SRC]BOOTDRIVR/
```

0037 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY